14. Kleine beeldschermen

Voor elke toepassing hebben we één of andere vorm van uitvoer nodig. We willen weten wat ons programma doet en wat de resultaten zijn. Soms is een LED als uitvoer voldoende en soms gebruiken we Thonny voor de uitvoer. Controller-toepassingen werken meestal autonoom, dan is een klein scherm aan het bordje toch veel handiger. We hebben al 7-segment displays gebruikt maar die zijn maar beperkt bruikbaar. Er zijn betere systemen die ook tekst en afbeeldingen weergeven.

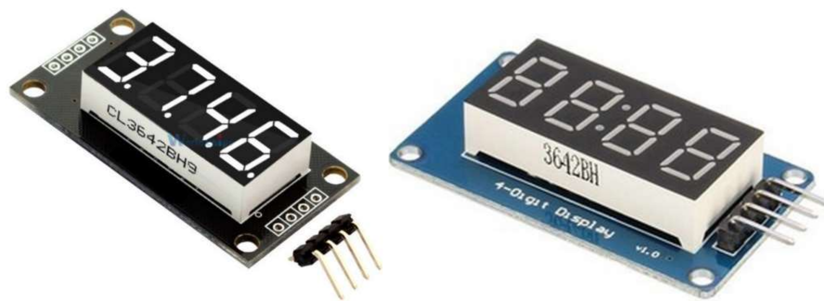
De beeldschermen van dit hoofdstuk verwachten meestal een string als input. Wij moeten die string opbouwen met stukken tekst, getallen en/of waarden van variabelen. Daarvoor gebruiken we de constructie van een string uit het hoofdstuk *List, Tuple en String*.

7-segment displays

7-segmentdisplays hebben we gezien in hoofdstuk *Experimenten met GPIO-poorten.* Elk cijfer bestaat uit 7 LED's en een LED voor het decimale punt. Voor een module met 4 cijfers moeten we 32 LED's aansturen. De bekabeling en de software worden erg ingewikkeld, dit is practisch ondoenbaar. Gelukkig bestaan er IC's met een seriële aansluiting waarmee een rij 7-segment displays aan te sturen zijn.

Aansturing met de TM1637

Met de TM1637 kan je 7-segment displays aansturen met maximaal 6 cijfers. Goed verkrijgbare modules met dit IC hebben 4 cijfers.



Figuur 78: TM1637-display

De eerste heeft decimale punten, de tweede heeft een dubbelpunt tussen het 2de en de 3de cijfer. Softwarematig gedraagt het dubbelpunt zich als een decimaal punt van het tweede cijfer. Het linker display is geschikt voor allerlei numerieke toepassingen,

het rechter voor een klok. De software voorziet ook in tekstweergave maar daar zijn ze echt niet voor geschikt.

De module heeft 4 aansluitingen: CLK, DIO, Vcc en GND. Vcc en GND sluit je aan op op resp. Vcc en GND van de controller. In het demo-programma is CLK verbonden met GPIO8 en DIO met GPIO7. De aansluitingen en het communicatieprotocol lijken op I2C, maar is er niet mee compatibel.

Voor het bordje bestaat er een library, die is te vinden op de Github-pagina van Mike Causer, https://github.com/mcauser/micropython-tm1637 . Hieronder is vind je een demo-programma. Kopieer bestand tm1637.py naar de controller en probeer het voorbeeld uit.

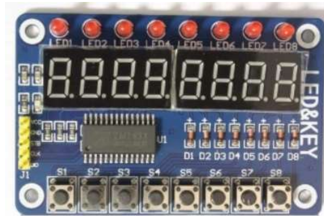
```
#-----------------------------#
#  tm1637-vb.py               #
#                             #
#  6 januari 2022             #
#  Dirk Ghysels               #
#-----------------------------#
from machine import Pin
import tm1637
from utime import sleep
mydisplay = tm1637.TM1637(clk=Pin(8), dio=Pin(7))
tekst = "1234567890"
mydisplay.show(tekst)
sleep(1)
temp = 36
mydisplay.temperature(temp)
sleep(1)
mydisplay.scroll(tekst,250)
sleep(1)
mydisplay.number(5739)
sleep(1)
mydisplay.numbers(12,13)
sleep(1)
```

.

- De temperatuur wordt weergegeven als 36°C. Alleen positieve getallen kleiner dan 100 zijn mogelijk.
- Methode number toont alleen getallen tussen -999 en 9999.
- Methode numbers toont alleen getallen tussen -9 en 99. Het resultaat wordt 12:13 of 12.13, naargelang het display.
- Het programma werkt op een RP2040, een ESP32, een ESP8266 en een SAMD21.
- De SAMD21 versie heeft een probleem: de library gebruikt functies min en max. MicroPython voor de SAMD21 kent die niet. Laat voor deze controller de 3 regels in de library weg die deze functies gebruiken

Aansturing met de TM1638

Met een TM1638 kan je 7-segment displays aansturen met 8 cijfers, 8 LED's en 8 schakelaars. De module hieronder wordt door verschillende fabrikanten geleverd.



Figuur 79: 7-segment display met tm1638

Mike Causer heeft hiervoor ook een library gemaakt, zie

https://github.com/mcauser/micropython-tm1638

Sluit het display aan op de controller: Vcc met Vcc, GND met GND, STB, CLK en DIO met de pinnen die in het voorbeeld gedefineerd zijn. Kopieer bestand tm1638.py naar de controller en test het voorbeeldprogramma. Het programma is getest met een RP2040 bordje en met een esp32. Het werkt niet met een SAMD21

```
#-----------------------------#
#  tm1638-vb1.py              #
#                             #
#  11 december 2021           #
#  Dirk Ghysels               #
#-----------------------------#

import tm1638, utime
from machine import Pin

# voor RP2040 en esp32
#tm = tm1638.TM1638(stb=Pin(28), clk=Pin(27), dio=Pin(26))
tm = tm1638.TM1638(stb=Pin(25), clk=Pin(26), dio=Pin(27))

while True:
    tm.leds(0b01010101)     # alle even LED's aan (tel vanaf
rechts)
    utime.sleep(.5)

    tm.leds(0)              # alle even LED's uit
    utime.sleep(.5)

    tm.show('Error')        # toon tekst
    utime.sleep(.5)

    tm.show('3.1419527')    # toon tekst
    utime.sleep(.5)

    tm.number(-1234567)     # toon geheel getal
    utime.sleep(.5)

    tm.clear()
```

```
    tm.temperature(23)      # toon temperatuur
    utime.sleep(.5)

    tm.humidity(56)         # toon relatieve vochtigheid
    utime.sleep(.5)

    tm.hex(0xdeadbeef)      # toon hexadecimaal getal
    utime.sleep(.5)

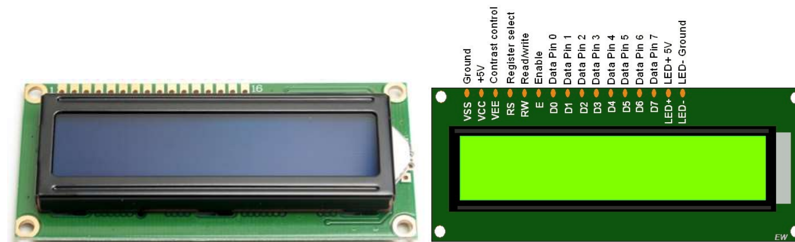
    tm.brightness(0)        # dim LEDs and cijfers

    tm.clear()              # dim alles

    print (tm.keys())       # welke toetsen zijn ingedrukt
```

LCD karakter-displays

Een karakter-display kan een vast aantal karakters (letters, cijfers, speciale tekens, …) weergeven. Er bestaan LCD[3]'s met 1 rij van 8 tekens tot LCD's met 4 rijen van 20 tekens.



Figuur 80: LCD, 2 rijen, 16 kolommen

Je ziet aan het aansluitschema hierboven dat er nogal wat verbindingen nodig zijn. Voor het Pico bordje is dat geen groot probleem, die heeft genoeg GPIO's. De bekabeling is vrij lastig en de kans op fouten vrij groot. Een betere oplossing is de LCD met een I2C aansluiting.

LCD met I²C-interface

Er bestaan LCD's met I2C convertor. Deze converteert de I2C -signalen naar de LCD. De convertor wordt aangesloten op de RPI met 4 draadjes: VCC, GND, SDA en SCL. De convertors zijn los te koop maar je vindt de LCD ook met gemonteerde convertor.

[3] LCD: Liquid Crystal Display



Figuur 81: LCD met I2C-convertor

Let op: het zijn 5V-bordjes maar ze kunnen zonder probleem rechtstreeks aangesloten worden op de GPIO-pinnen. De data-communicatie verloopt in één richting, van de controller naar de LCD. De LCD zal nooit gevaarlijk hoge spanningen op een GPIO zetten en de LCD kan de lage signaalniveaus van de controller wel decoderen. De LCD heeft 4 aansluitpinnen die moet je verbinden met de RPI volgens volgende tabel:

	LCD-nr	Controller
1	GND	GND
2	VCC	**5 V**
3	SDA	Sda
4	SCL	Scl

De 5 volt voor de LCD betrek je uit aansluiting VBUS van het bordje. Controleer alles programma lcd-scan uit de paragraaf over I2C. Je zou moeten zien dat I2C-adres 27 in gebruik is, dat is het adres van de LCD. Kijk de bedrading nog eens na als je 27 niet ziet.

Voor de aansturing van de LCD kan je de software van Tyler Peppy gebruiken. Die staat op zijn Gkithub pagina, op https://github.com/T-622/RPI-PICO-I2C-LCD . Je vindt er ook een demo-programma waarmee je de werking test en waarmee je kennis maakt met de mogelijkheden van de software. De library bestaat uit bestanden pico_i2c_lcd.py en lcd_api.py. Kopieer deze bestanden naar het pico-bordje. Het voorbeeld hier onder toont hoe je informatie op het scherm zet. Dat kan alleen met functie putstr(). Die zet een string op de LCD. Voor we de tweede string op het scherm zetten, plaatsen we de cursor op naar plaats (0,1) met functie move_to. Dat is op de,nulde kolom, eerste rij. (we tellen vanaf 0).

```
#-----------------------------#
#  hello-lcd-i2c.py           #
#                             #
#  5 oktober 2021             #
#  Dirk Ghysels               #
#-----------------------------#
import machine, utime
from pico_i2c_lcd import I2cLcd
```

```
from lcd_api import LcdApi
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16

i2c = machine.I2C(0,sda=machine.Pin(24), scl=machine.Pin(25),
freq=400000)
lcd = I2cLcd(i2c, 0x27, I2C_NUM_ROWS, I2C_NUM_COLS)
lcd.clear()
naam1 = " Jef Neve  "
naam2 = " Bert Joris "
lcd.putstr(naam1)
lcd.move_to(0,1)
lcd.putstr(naam2)
```



Figuur 82: LCD scherm

Op de convertor vind je een instelweerstand om de helderheid van de LCD in te stellen. Draai er eens aan als je niets ziet op het scherm. Het programma werkt ook voor kleinere of grotere schermen: verander de waarde van I2C_NUM_ROWS en I2C_NUM_COLS

Het volgende voorbeeld laat zien hoe je een klok met LCD maakt. De klok gebruikt de RTC en zet gegevens op het scherm. Het gelijkzetten van de klok moet je nog zelf toevoegen.

```
#-----------------------------#
#  lcd-klok.py                #
#                             #
#  26 maart 2021              #
#  Dirk Ghysels               #
#-----------------------------#
import machine, utime
from machine import Pin

from pico_i2c_lcd import I2cLcd
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16
```

```
i2c = machine.I2C(0,sda=machine.Pin(24), scl=machine.Pin(25), freq=400000)
lcd = I2cLcd(i2c, 0x27, I2C_NUM_ROWS, I2C_NUM_COLS)
lcd.clear()
tijdStr = ""
datumStr = ""
sec0 = 0
dag0 = 0

while(True):
    time = utime.localtime()
    uur = time[3]
    min = time[4]
    sec = time[5]
    dag = time[2]
    maand = time[1]
    jaar = time[0]
    if (sec0!=sec):
        sec0=sec
        lcd.move_to(0,0)
        tijdStr = "%02u:%02u:%02u" %(uur, min, sec)
        datumStr = "%02u/%02u/%4u" %(dag, maand, jaar)
        lcd.putstr(tijdStr)
        lcd.move_to(0,1)
        lcd.putstr (datumStr)
        utime.sleep(.1)
```

OLED-displays

Kleine monochrome OLED-displays zijn goedkoop, energiezuinig en hebben een zeer goede beeldkwaliteit. In Elektor van januari 2018 vind je een meer informatie over OLED. Ze hebben een I2C-interface en worden bijna allemaal aangestuurd door dezelfde chip, de SSD1306. Je vindt een heleboel van die displays, met prijzen vanaf 2,50 euro. Deze schermpjes zijn klein, maar door de scherpte van het beeld krijg je er toch veel informatie op.



Figuur 83: OLED-displays

Er bestaan 2-kleuren OLED's maar stel je daar niet te veel van voor. Het bovenste stukje van het scherm is altijd geel, de rest altijd blauw. De bordjes werken met 3,3 V, sluit ze niet aan op de 5V voeding. Een bruikbaar type is een OLED met diameter 0.96 inch en een resolutie van 128x64. Er zijn er nog kleinere maar 96 inch is zeker klein genoeg.

Verbind Vin met de 3,3V van Pico-bordje, GND met GND en sluit sla en scl aan aan de I2C-poort van de controller. Controleer alles met programma i2c-scan.py. Hier zou je moeten zien dat I2C-adres 0x3c (hexadecimale vorm) of 60 (10-delig) in gebruik is, dat is het adres van de OLED. Kijk de bedrading nog eens na als je 0x3c niet ziet.

Ssd1306 voor ESP8285 en ESP8266

MicroPython heeft alleen voor de EPS8285 en ESP8266 een module die een OLED display aanstuurt. Het voorbeeld hieronder toont het gebruik er van. Methode tekst zet tekst op het scherm:

display.tekst("abc", a, b)

De tekst "abc" staat op a pixels van de bovenrand en b pixels van de linkerrand. De tekst wordt opgeslagen in een schermbuffer, display.show() zet de informatie van de buffer op het scherm.

Voor numerieke uitvoer moeten we converteren naar een string.

```
#------------------------------#
#  ssd1306-esp.py              #
#  esp8266 en esp8285          #
#                              #
#  25 december 2021            #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, I2C, SoftI2C
import ssd1306
i2c = I2C(scl=Pin(5), sda=Pin(4), freq = 100000)
print(i2c.scan())
breedte  = 128        # schermbreedte in pixel
hoogte = 64           # schermhoogte in pixel

display = ssd1306.SSD1306_I2C(breedte, hoogte, i2c)
display.fill(0)
display.show()
display.text('Hello World', 1, 1)
display.text('25 december 2021', 1, 20)
k=2342
display.text(str(k),45,40)
display.show()
```

De scherminformatie wordt opgeslagen in een schermbuffer. **display.text(string, a, b)** zet *string* met schermcoördinaten (a, b) in de buffer, oled.show() zet de informatie van de buffer op het scherm.

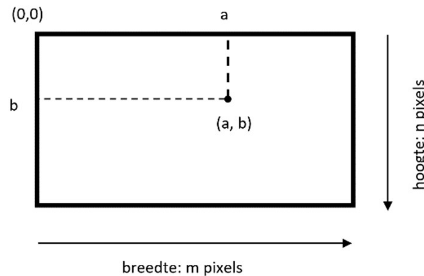
Andere basisfuncties zijn:

- **display.poweroff()** schakelt het scherm uit
- **display.poweron()** schakelt het scherm aan
- **display.contrast(0)** dimt het scherm
- **display.contrast(255):** maximale helderheid
- **display.invert(True)** roteert het scherm 180°
- **display.invert(False):** geen rotatie

Verander, indien nodig parameters breedte en hoogte. Die informatie vind je bij de gegevens van het display.

Tekenen op een OLED-scherm

Een OLED scherm bestaat uit punten die we pixels noemen. Elke pixel is apart aanstuurbaar, we kunnen zo tekeningen maken op een OLED-scherm. Een pixel krijgt schermcoördinaten (a,b), a is het aantal pixels tot de linkerrand, b tot de bovenrand.



Figuur 84: schermcoördinaten

Hieronder zie je van enkele populaire schermen de afmetingen. De diagonaal is gemeten in inches, hoogte en breedte in pixels.

Diagonaal	Breedte	Hoogte
0.42”	72	40
0.66”	64	48
0.96”	128	64
1.3 “	128	64
0.91”	128	32

De ssd1306 library van MicroPython heeft enkele grafische functies:

- **display.fill(0)** vult het scherm met kleur 0
- **display.pixel(0, 10)** tekent een punt op plaats (0,10)
- **display.pixel(0, 10,1)** tekent het punt met kleur 1
- **display.hline(0,8,4,1)** tekent een horizontale lijn door (0,8) met dikte 4 en kleur 1
- **display. vline(0,8,4,1)** tekent een vertikale lijn
- **display.line(0,0,127,63,1)** tekent een lijn van (0,0) naar (127,63)
- **display.rect(0,10,12,30,1)** tekent een rechthoek met hoekpunten (0,10) en (12,30) en kleur 1
- **display.fill_rect(0,10,12,30,1)** tekent een opgrvulde rechthoek

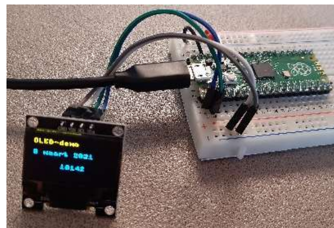
Ssd1306 voor andere controllers

Op website http://electronoobs.com/eng_arduino_tut138.php vind je een library voor OLED’s die je voor alle andere controllers kunt gebruiken. Kopieer hieruit

bestand ssd1306.py naar de controller. Van het voorbeeldprogramma hierboven moet je alleen de definitie van i2c aanpassen: andere controllers gebruiken soms andere poorten voor de I2C-bus. Bij electronoobs vind je een uitgebreider voorbeeld. Daar zie je ook grafische mogelijkheden van het scherm. Hieronder zie je de versie voor een RP2040:

```
#-----------------------------#
#  ssd1306-rp2040.py          #
#  voor RP2040                #
#                             #
#  25 december 2021           #
#  Dirk Ghysels               #
#-----------------------------#
from machine import Pin, I2C, SoftI2C
import ssd1306
i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq = 100000)
print(i2c.scan())
breedte  = 128        # schermbreedte in pixel
hoogte = 64           # schermhoogte in pixel

display = ssd1306.SSD1306_I2C(breedte, hoogte, i2c)
display.fill(0)
display.show()
display.text('Hello World', 1, 1)
display.text('25 december 2021', 1, 20)
k=2342
display.text(str(k),45,40)
display.show()
```



Figuur 85: een klein OLED-scherm

Adafruit heeft een grafische library voor grafische schermen. Je vindt die op

https://github.com/adafruit/micropython-adafruit-gfx

Kopieer hieruit bestand gfx.py naar de controller. Zoals library ssd1306.py gaan we grafische elementen definiëren in de schermbuffer. De buffer zetten we op het scherm met instructie

oled.show()

De initiatie van het grafisch deel is

import gfx
graphics = gfx.GFX(breedte, hoogte, oled.pixel)

breedte en hoogte zijn schermbreedte en hoogte, oled is een gedefinieerd display.

Enkele functies uit de library zijn:

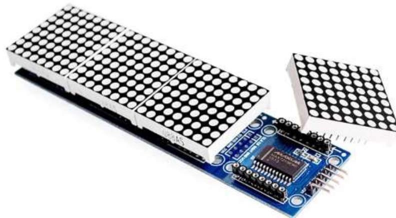
- oled.fill(0): maakt het volledige scherm zwart
- oled.fill(1): maakt het volledige scherm wit
- graphics.line(a,b,c,d,1): tekent een lijn van punt (a,b) naar (c,d)
- graphics.rect(a,b,c,d,1): tekent een rechthoek met linkerbovenhoek (a,b) en rechterbenedenhoek (c,d)
- graphics.fill_rect(a,b,c,d,1): vult de rechthoek op
- graphics.circle(a,b,r,1): tekent een cirkel met middelpunt (a,b) en straal r
- graphics.fill_circle(a,b,r,1): vult de cirkel op
- graphics.triangle(a,b,c,d,e,f,1): tekent een driehoek met hoeken (a,b), (c,d), en (e,f)
- graphics.fill_triangle(a,b,c,d,e,f,1): vult de driehoek op

Voorbeeld:

```
breedte = 128
hoogte = 64
from machine import Pin, I2C, SoftI2C
import ssd1306, gfx
from time import sleep
i2c = I2C(0,scl=Pin(17), sda=Pin(16), freq = 200000)
print(i2c.scan())
oled = ssd1306.SSD1306_I2C(breedte, hoogte, i2c)
graphics = gfx.GFX(breedte, hoogte, oled.pixel)
while True:
    oled.fill(0)
    graphics.line(0,0,100,200,1)
    graphics.circle(60,50,20,1)
    graphics.rect(10,10,100,30,1)
    oled.show()
    sleep (1)
    graphics.fill_circle(60,50,20,1)
    graphics.fill_rect(10,10,100,30,1)
    oled.show()
    sleep (1)
```

Dot matrix displays

Deze displays bestaan uit een matrix van 64 LED’s, verdeeld over 8 rijen en 8 kolommen. Ze worden aangestuurd door IC MAX7219. Ze hebben 2 SPI poorten: een ingang en een uitgang om ze te koppelen. Ze worden los verkocht of in modules van meerdere elementen.



Figuur 86: 4-voudig dot-matrix display

Een library om het display aan te sturen vinden we op de Github: https://github.com/mcauser/micropython-max7219. Kopieer uit die pagina bestand max7219.py naar de controller. Het voorbeeldprogramma hieronder is gebaseerd op de voorbeelden en aangepast voor de RP2040.

Eerst definiëren we een SPI-poort:

spi = machine.SPI(0, baudrate= 10000000, sck = psck, mosi = pmosi, miso = pmiso)

- Parameter 0 is het nummer van de SPI-poort.
- De baudrate is een maat voor de communicatiesnelheid. 1000000 lijkt een goede waarde te zijn.
- De drie volgende parameters geven aan welke pinnen we gebruiken voor sck, mosi en miso.

De tweede constructor definieert het scherm.

screen = max7219.Max7219(32, 8, spi, pss)

- De eerste parameter is het aantal kolommen van de volledige module, 4 displays met 8 kolommen.
- De tweede parameter is het aantal rijen: 8 LED's onder elkaar.
- spi is de SPI die we hierboven gedefinieerd hebben.
- De laatste parameter is het SPI-SS signaal voor de module. Elk apparaat op een SPI-bus heeft zijn eigen SS.

Methode tekst() zet een string in de schermbuffer. De drie laatste parameters zorgen voor de uitlijning van de letters. Methode show() zet de buffer op het scherm. Verbind de ingang van de module met de controller. In het programma zie je welke pinnen je moet gebruiken.

```
#-----------------------------#
#  max7219-demo.py            #
#                             #
#  12 maart 2021              #
#  Dirk Ghysels               #
#-----------------------------#
```

```
from machine import Pin, SPI
import max7219, utime
psck = Pin(2)
pmosi = Pin(3)
pmiso = Pin(4)
pss = Pin(5)
spi = machine.SPI(0, baudrate= 10000000, sck = psck, mosi = pmosi, miso = pmiso)
screen = max7219.Max7219(32, 8, spi, pss)
screen.text('Beau',0,1,1)
screen.show()
utime.sleep(1)
screen.fill(0)
screen.text('Annemie',0,1,1)
screen.show()
```